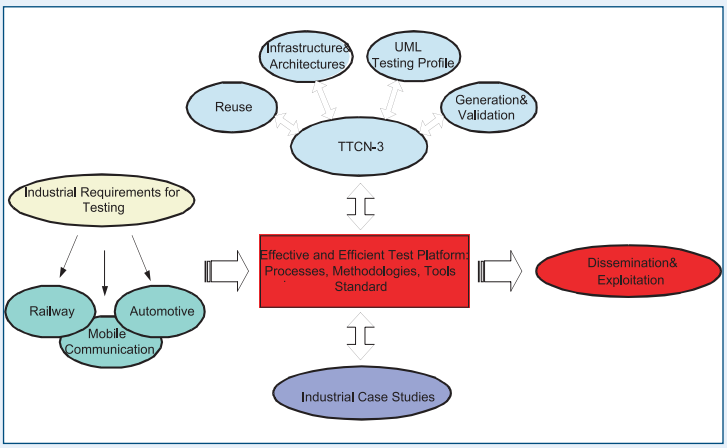

Topics and goals within TT-Medal.

studies, and dissemination and standardisation work. There are strong relationships such as major requirements from the industrial cases and theories from methodology that are used as input for the tool-development work packages. Since several partners are actively involved in standardisation work at ETSI and OMG, there is also an exchange of ideas with the originators of TTCN-3 and UML2.0.

The major outcome of the TT-Medal project is an effective and efficient test platform. The platform combines the work of the five different work packages and provides necessary processes, methodologies, tools and standards to improve testing. The effectiveness of the platform is indicated by better product quality, and efficiency ensures that this quality is achieved within a given time frame and resource limits. The TT-Medal project will also show how these new technologies should be used by providing training and organising workshops, as well as demonstrating where and why TT-Medal technologies should be used by gathering industrial experience. The figure below summarises the work and the major elements of the TT-Medal project.

The TT-Medal project, which runs until end of 2005, may be followed in various ways: participate in our upcoming national and international workshops and exhibitions, visit our project Web site for the latest news and publications, or contact a partner or national representative from the project for further information.

Links:
Project Web site: http://www.tt-medal.org

Testing Languages Web sites:
http://www.etsi.org/ptcc/ptccttcn3.htm
http://www.fokus.fraunhofer.de/tip/projects/u2tp

Please contact:
Wan Fokkink, CWI, The Netherlands
Tel: +31 20 592 4104
E-mail: wan.fokkink@cwi.nl

Matti Kärki, VTT Electronics, Finland
Tel: +358 8 551 2268
E-mail: matti.karki@vtt.fi

Ina Schieferdecker
Fraunhofer FOKUS, Germany
Tel: +49 30 3463 7241
E-mail: schieferdecker@fokus.fraunhofer.de

Automated Verification of Groupware Protocols

by Maurice ter Beek, Mieke Massink, Diego Latella, Stefania Gnesi, Alessandro Forghieri and Maurizio Sebastianis

Recently, researchers from the Formal Methods and Tools (FM&&T) group of ISTI-CNR teamed up with researchers from think3, a global provider of integrated product development solutions. The goal was to apply formal modelling and verification techniques to enhance think3's Product Data Management (PDM) application, 'thinkteam' (a registered trademark of think3 Inc) with a publish/subscribe notification service.

Computer-Supported Cooperative Work (CSCW) is an interdisciplinary research field which deals with understanding how people work together and finding ways in which computer technology can assist. This technology mostly consists of multi-user computer systems called groupware (systems). Many concepts and techniques from computer science, like concurrency control and data consistency, need to be rethought in the groupware domain. This has led to the development of new formal models like team automata, which were introduced explicitly for the description and analysis of groupware systems (see [TA] for details). Groupware systems are typically classified according to two dichotomies, namely (i) whether the users are working together at the same time (synchronous) or at different times (asynchronous) and (ii) whether they are working together in the same place (co-located) or in different places (dispersed). Examples of synchronous groupware include video conferencing and multi-user games, while electronic mail and the version-control systems which are often used in software engineering to coordinate changes made by multiple programmers to the same program are examples of asynchronous groupware.

An additional difficulty that arises during the design of synchronous group-

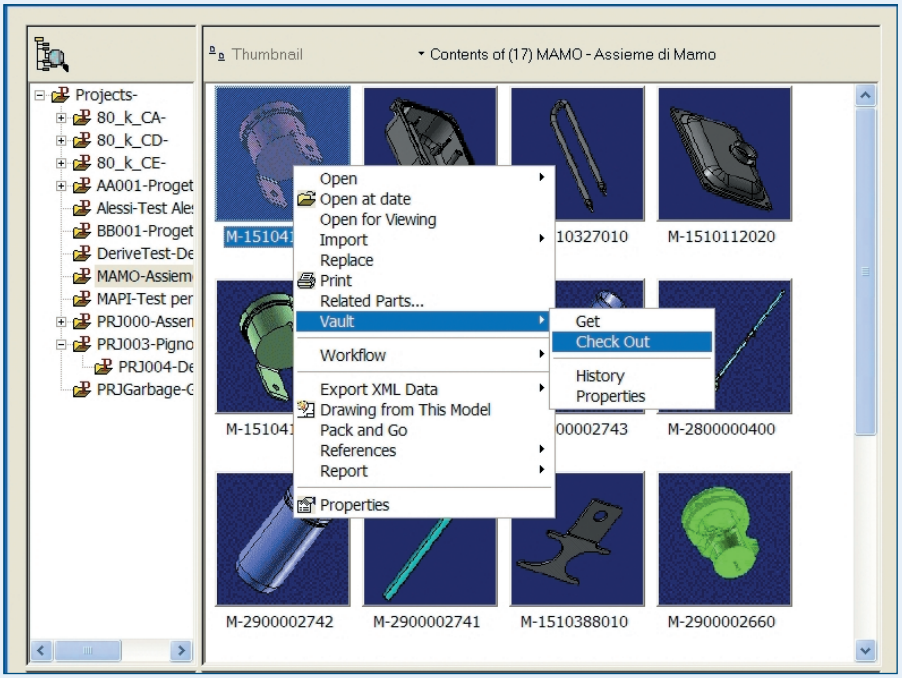

Figure 1: A thinkteam user checks out a document from the vault.

ware is the inherently distributed nature of such systems, which makes it necessary to address issues like network communication, concurrency control and distributed notification. This has led to the development of groupware toolkits that aid developers with programming abstractions aimed at simplifying the development of groupware applications.

We are currently modelling and verifying the addition of a publish/subscribe notification service to thinkteam, think3's PDM application catering to the product/document management needs for design processes in the manufacturing industry. The main strengths of this service are a rapid deployment and start-up cycle, flexibility, and a seamless integration with thinkdesign - think3's CAD solution - as well as with other third party products. Thinkteam allows enterprises to capture, organise, automate, and share engineering product information in an efficient way. The controlled storage and retrieval of document data in PDM applications is traditionally called vaulting, where the vault is a file-system-like repository. Its two main functions are (i) to provide a single, secure and controlled storage environment in which the documents controlled by the PDM application are managed, and (ii) to prevent inconsistent updates or changes to the document base while still allowing the maximal level of access compatible with the business rules. While the first function is integrated in the lower layers of the vaulting system, the second function is implemented in thinkteam's underlying groupware protocol by a standard set of operations on the vault, namely get (extract a read-only copy of a document), import (insert an external document), checkout (extract an exclusive copy of a document with the intent of modifying it), checkin (replace an edited and previously checked-out document), checkinout (replace an edited document in the vault, while at the same time retaining it as checked out), and uncheckout (cancel the effects of a checkout). Figure 1 shows a screen shot of thinkteam.

Thinkteam is thus an example of an asynchronous and dispersed groupware system. In order to develop the publish/subscribe notification service we first had to define an abstract specification (model) of the groupware protocol underlying thinkteam. The idea behind this type of service is to more widely inform the users of an application by intelligent data sharing. For instance, whenever a user publishes a file by sending it to a centralised repository, all users who subscribe to that file are notified automatically via a multicast communication. The automatic verification of publish/subscribe notification services is currently attracting a lot of attention. The addition of such a service to thinkteam should allow us to solve a problem that commonly arises in connection with the use of composite documents and which is a variant of the classic 'lost update' phenomenon, depicted in Figure 2. This phenomenon arises when a user performs a checkout/modify/checkin cycle on a document that may be used as reference copy by other users.

Our research shows that with relatively simple models we can verify highly relevant properties of groupware protocols with freely available verification tools. Model checking is an automatic technique to verify whether a system design satisfies its specifications. The verification is exhaustive, ie all possible input combinations and states are taken into account. One of the best-known and most successful model checkers is Spin, which was developed at Bell Labs during the last two decades. It offers a spectrum of verification techniques, is freely available, and is very well documented. Publish/subscribe systems require specific properties, including data consistency through

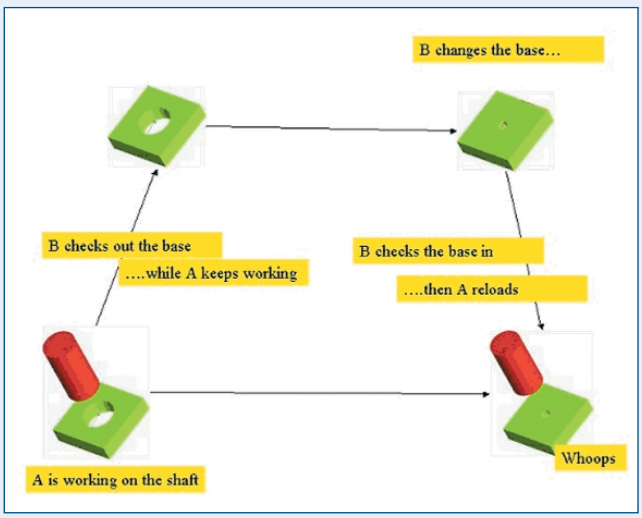

Figure 2: The 'lost update' phenomenon.

distributed notification, view consistency, absence of (user) starvation and key issues related to concurrency control [BMLG]. The properties we verify are mostly formalised as formulae of a Linear Temporal Logic (LTL). This has the advantage that they are close to the 'scenarios' design. LTL formulae reflect properties of typical behaviour (or uses) of the groupware system. This makes it possible to evaluate design alternatives before their implementation and validation and not only afterwards. Design errors constitute up to 40% of software errors and are among the most expensive to resolve when discovered only after the implementation phase. Our system allows these to be detected early on in development, leading to considerable reductions in cost and improvements in quality. The specification (model) of thinkteam's underlying groupware protocols has been developed using this methodology and in close collaboration between members of FM&&T and think3.

This research is conducted within the framework of a broader national research project on global computing, funded by the Italian Ministry of Research and Education (MIUR/SP4).

Links:
[TA]: http://fmt.isti.cnr.it/~mtbeek/TA.html
[BMLG]: http://fmt.isti.cnr.it/WEBPAPER/TR-61.pdf
FM&&T: http://fmt.isti.cnr.it/
think3: http://www.think3.com/
Spin: http://www.spinroot.com/

Please contact:
Maurice ter Beek, Mieke Massink
ISTI-CNR, Italy
E-mail: {terbeek,massink}@isti.cnr.it

Parallel Model-Checking

by Luboš Brim

With the rapid increase in computer system complexity, it has become very important to develop formal methods to ensure their quality. Several novel parallel and distributed techniques for enumerative model-checking of safety and liveness properties expressed in a simple temporal logic have been developed in the Parallel and Distributed Systems laboratory at the Masaryk University Brno under the project 'Automated Verification of Parallel and Distributed Systems'.

Early detection of errors requires application of advanced analysis, verification and validation techniques to test modelling resources, temporal properties, data-type invariants, and security properties. Various techniques for automated and semi-automated analysis and verification of computer systems have been proposed. In particular, model checking has become a very practical technique due to its push-button character. The basic principle behind model checking is to build a finite model of the system under consideration together with a formal description of the verified property in a suitable temporal logic. The model-checking algorithm is a decision procedure, which in addition to the yes/no answer returns a trace of a faulty behaviour in cases where the checked property is not satisfied by the model. One of the additional advantages of this approach is that verification can be performed against partial specifications, by considering only a subset of all specification requirements. This allows for increased efficiency by checking correctness with respect to only the most relevant requirements.

Although model checking has been applied fairly successfully for verification of quite a few real-life systems, its applicability to a wider class of practical systems has been hampered by the so-called state explosion problem (ie the enormous increase in the size of the state space). For large industrial models, the state space does not completely fit into the main memory of a computer. Consequently, when the memory becomes exhausted and the system starts swapping, the model-checking algorithm becomes very slow.

A typical approach to dealing with these practical limitations is to increase the computational power (especially random-access memory) by building a powerful parallel computer as a network (cluster) of workstations. Individual workstations communicate through a message-passing interface such as MPI. From outside a cluster it appears as a single parallel computer with high computing power and a huge amount of memory. In recent years a lot of effort has been put into using parallel and distributed environments to solve the computational and space complexity bottlenecks in model-checking systems.

The main question is whether we can (re)design verification techniques in such a way that they could be implemented on parallel computer architectures. In other words, we would like to find techniques for decomposing complex verification problems into 'smaller' independent sub-tasks, which could then be either further decomposed or directly solved. Efficient parallel solution of many problems often requires the invention of original, novel approaches that are radically different from those used to solve the same problems sequentially. Several methods for parallel model-checking have been accepted and implemented in industrial tools. Performance results on either parallel machines or on a cluster of workstations show significant improvements over sequential techniques, both in extension of the size of the problem and in computational times, along with adequate scalability with the number of processors.

We have developed several novel parallel and distributed techniques for